IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Inventor(s): Ricardo L. de Queiroz
Application No.: 10/037,905
Filed: 11/09/2001
Examiner: S. Brinich
Art Unit: 2624
Title: BLOCKING SIGNATURE DETECTION FOR IDENTIFICATION OF JPEG IMAGES

CERTIFICATE OF MAILING
I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Assistant Commissioner for Patents, Washington, D. C. 20231, on
Dec. 10, 2003
(Date of deposit)

Mark Z. Dudley
(Name of applicant, assignee, or Registered Representative)
M Z Dudley
(Signature)
Dec. 9, 2003
Date of Signature

Assistant Commissioner for Patents
Washington, D.C. 20231

Sir:

APPEAL BRIEF

In consideration of the Notice of Appeal From the Examiner's Decision, filed by appellant on 10-10-2003, appellant submits this Appeal Brief.

Authorization is hereby granted to charge the appeal brief fee under 37 CFR 1.17(f) to deposit account 24-0025 (two additional copies of this authorization is enclosed herewith.)

```
01/05/2004 TACREE   00000002 240025   10037905
01 FC:1402       330.00 DA
```

TABLE OF CONTENTS

PAGE

TABLE OF CASES

REAL PARTY IN INTEREST

The real party in interest in the present Appeal is Xerox Corporation, the assignee.

RELATED APPEALS AND INTERFERENCES

There are no other appeals or interferences known to appellants that will affect, be affected by, or have a bearing on the pending appeal.

STATUS OF CLAIMS

Claim 25 remains on appeal.

Claim 25 stands rejected under 35 USC 102(e).

STATUS OF AMENDMENTS

No amendment has been filed subsequent to final rejection.

SUMMARY OF THE INVENTION

As claimed in claim 25, the invention is directed to an analysis of blocking discontinuities in an image as being indicative of compression history. As was originally stated in the Specification at page 4, lines 14-18:

> 'Thus, discontinuities across block boundaries (blocking effects) account for the most noticeable compression artifact caused by JPEG compression. The higher the compression the higher the blocking. The present invention performs an analysis of the blocking discontinuities as an indicative of compression history."

ISSUES

Whether claim 25 is unpatentable under 35 USC 102(e) as anticipated by Hintzman et al. (US Patent No. 5,818,364).

GROUPING OF CLAIMS

Claim 25 of this patent application is the sole claim at issue in this Appeal.

ARGUMENT

(I) Rejection based on 35 USC 112, first paragraph

None applicable.

(II) Rejection based on 35 USC 112, second paragraph

None applicable.

(III) Rejection based on 35 USC 102(e)

Claim 25 stands rejected as being anticipated by Hintzman et al.

The Examiner noted that Applicant has already distinguished Hintzman et al. in Paper No. 2 (preliminary amendment filed on 11-09-01). In that amendment, Applicant amended claim 25 to more clearly denote the claimed detection of blocking artifacts in an image.

In support of the Final Rejection, the Examiner equates a "block boundary" with "a byte boundary", and equates "artifact" with "artificially introduced bits". The interpretation adopted by the Examiner is incorrect and was improperly used as the basis for the 102(e) rejection. The Examiner states that "it is not clear what recited feature of the 'blocking artifacts in the image indicative of compression' distinguishes from the byte boundary information detected in Hintzman et al." The pending claim clearly distinguishes over the cited art. The 102(e) rejection is not properly supported and therefore should be reversed.

A "block boundary" is not equivalent to the "byte boundary" taught in Hintzman. The byte boundary described in Hintzman et al. is described at col. 5, lines 11-14 in reference to Figure 2, and is denoted as being consistent with the data structure format standardized in the well-known JPEG data compression scheme. The Board is invited to review the following well-known publications, copies of which are appended to this Appeal Brief:

G. K. Wallace, "The JPEG Still Picture Compression Standard," *IEEE Trans. on Consumer Electronics*, vol. 38, no. 1, pp. 18-34, Feb. 1992.

Pennebaker & Mitchell, *JPEG: Still Image Data Compression Standard*, Van Nostrand Reinhold (1993), at Chapter 7.4, pp. 105-109.

Those skilled in the art would not equate the byte boundaries in a JPEG encoded bit stream with the block boundaries of an image. In fact, even Hintzman et al. refer to the "JPEG standard defined RESTART and END_OF_IMAGE ("EOI") markers" as being present on the compressed JPEG data stream on byte boundaries. These terms in Hintzman et al. simply refer to bits denoting positional relationships of bytes of data. These bits are used for data handling functions "to indicate on which byte boundary in the 32-bit input word the marker was detected."

During the phase in which a bit stream represents a compressed (encoded) image, and thus the image is available for storage in memory, the actual source image, and any artifacts due to such compression, are abstractions. The block boundaries in a JPEG encoding scheme are represented by a variable amount of bits which are not byte-aligned, in general, and thus byte boundary information is irrelevant to the image blocks (and the image block artifacts) that may exist, abstractly, in a JPEG-encoded image data stream. Accordingly, the "byte boundaries" disclosed in Hintzman et al. have insufficient relation to the "block boundaries" claimed in claim 25 such that one skilled in the art would consider them the same, or similar.

The Examiner states that "it is not clear what recited feature of the 'blocking artifacts in the image indicative of compression' distinguishes from the byte boundary information detected in Hintzman et al." The Examiner also equates "artifact" with "artificially introduced bits" and seems to equate the byte boundary information of Hintzman et al. with the claimed blocking artifacts. As already stated, the byte boundaries disclosed in the cited reference have insufficient relation to the claimed block boundaries and blocking artifacts. An interpretation of these terms as being distinguishable fully comports with their usage provided in the cited reference, in the pending application, and in common usage by those skilled in the art. The term "artifact" in conventional usage refers to an image defect in a compressed image that was not present in the original, uncompressed source image. An image artifact is typically regarded as something "wrong" with the picture from a visual standpoint. As was originally stated in the Specification at page 4, lines 14-18:

> 'Thus, discontinuities across block boundaries (blocking effects) account for the most noticeable compression artifact caused by JPEG compression. The higher the compression the higher the blocking. The present invention performs an analysis of the blocking discontinuities as an indicative of compression history."

Claim 25 recites: "...blocking artifacts presented in the form of discontinuities across *block boundaries in the image...*" (emphasis added.) The block boundaries are where such discontinuities in the decoded image are typically noticeable, and according to the claimed invention, the presence of these discontinuities in their abstract form (i.e., in the encoded data stream) may be detected, and the presence of such discontinuities will indicate that the image has been compressed.

Furthermore, the Examiner supports the rejection upon alleged disclosure of the second claimed step in claim 25, that is, alleged disclosure of "providing an output indicative of compression upon detection of the blocking artifacts." The Examiner presumably refers to Hintzman et al. at col. 5, line 24 (production of a 4-bit output in response to detection of a marker by detector 205.) These "4 bits are output to indicate on which byte boundary in the 32-bit input word the marker was detected." Hintzman et al. teaches only an output that denotes the location of a marker, and there is no description or suggestion therein of an output indicating image compression, nor is there an output provided in response to a detection of blocking artifacts.

Hintzman et al. merely discloses the detection of byte boundary information in a Huffman code word coefficient pair by use of a 4-bit string on bus 202 to a RLL detector 203. (See col. 2, lines 6-10; col. 4, lines 17-22.) There is no disclosure or teaching of the claimed detection of blocking artifacts in an image, with such artifacts being detected for the purpose of determining whether compression has been done, with a response being the provision of an output that is indicative of such compression.

Anticipation occurs only if a single prior art reference contains each and every element of the patent at issue, operating in the same fashion to perform the identical function as the patented product. Any degree of physical difference between the patented product and the prior art, no matter how slight, defeats the claim of anticipation. (*American Permahedge, Inc., v. Barcana, Inc.* , 857 F. Supp. 308, 32 USPQ2d 1801, 1807-08 (S.D.N.Y. 1994)). The Examiner has failed to show the anticipation of each and every element of claim 25 in Hintzman et al., and thus has failed to make a prima facie showing of anticipation.

(IV) Rejection based on 35 USC 103

None applicable.

(V). Other Rejections

None applicable.

The Board of Appeals is respectfully urged to reverse the Examiner's rejection of claim 25 and allow the application in its present form.

Respectfully submitted,

M Z Dudley

Mark Z. Dudley
Attorney for Appellant
Registration No. 33,110
(716) 265-7014

MZD
12-10-03

Xerox Corporation
Xerox Square 20A
Rochester, New York 14644

APPENDIX I
(CLAIMS APPEALED)

25. (Amended) A method to detect if an image is compressed, comprising the steps of:

(a) detecting blocking artifacts presented in the form of discontinuities across block boundaries in the image, said blocking artifacts thereby being indicative of compression; and

(b) providing an output indicative of compression in response to the detection of the blocking artifacts.

APPENDIX II

Attached is a copy of G. K. Wallace, "The JPEG Still Picture Compression Standard," *IEEE Trans. on Consumer Electronics*, vol. 38, no. 1, pp. 18-34, Feb. 1992.

The JPEG Still Picture Compression Standard

Gregory K. Wallace
Multimedia Engineering
Digital Equipment Corporation
Maynard, Massachusetts

Submitted in December 1991 for publication in IEEE Transactions on Consumer Electronics

This paper is a revised version of an article by the same title and author which appeared in the April 1991 issue of Communications of the ACM.

Abstract

For the past few years, a joint ISO/CCITT committee known as JPEG (Joint Photographic Experts Group) has been working to establish the first international compression standard for continuous-tone still images, both grayscale and color. JPEG's proposed standard aims to be generic, to support a wide variety of applications for continuous-tone images. To meet the differing needs of many applications, the JPEG standard includes two basic compression methods, each with various modes of operation. A DCT-based method is specified for "lossy" compression, and a predictive method for "lossless" compression. JPEG features a simple lossy technique known as the Baseline method, a subset of the other DCT-based modes of operation. The Baseline method has been by far the most widely implemented JPEG method to date, and is sufficient in its own right for a large number of applications. This article provides an overview of the JPEG standard, and focuses in detail on the Baseline method.

1 Introduction

Advances over the past decade in many aspects of digital technology - especially devices for image acquisition, data storage, and bitmapped printing and display - have brought about many applications of digital imaging. However, these applications tend to be specialized due to their relatively high cost. With the possible exception of facsimile, digital images are not commonplace in general-purpose computing systems the way text and geometric graphics are. The majority of modern business and consumer usage of photographs and other types of images takes place through more traditional analog means.

The key obstacle for many applications is the vast amount of data required to represent a digital image directly. A digitized version of a single, color picture at TV resolution contains on the order of one million bytes; 35mm resolution requires ten times that amount. Use of digital images often is not viable due to high storage or transmission costs, even when image capture and display devices are quite affordable.

Modern image compression technology offers a possible solution. State-of-the-art techniques can compress typical images from 1/10 to 1/50 their uncompressed size without visibly affecting image quality. But compression technology alone is not sufficient. For digital image applications involving storage or transmission to become widespread in today's marketplace, a standard image compression method is needed to enable interoperability of equipment from different manufacturers. The CCITT recommendation for today's ubiquitous Group 3 fax machines [17] is a dramatic example of how a standard compression method can enable an important image application. The Group 3 method, however, deals with bilevel images only and does not address photographic image compression.

For the past few years, a standardization effort known by the acronym JPEG, for Joint Photographic Experts Group, has been working toward establishing the first international digital image compression standard for continuous-tone (multilevel) still images, both grayscale and color. The "joint" in JPEG refers to a collaboration between CCITT and ISO. JPEG convenes officially as the ISO committee designated JTC1/SC2/WG10, but operates in close informal collaboration with CCITT SGVIII. JPEG will be both an ISO Standard and a CCITT Recommendation. The text of both will be identical.

Photovideotex, desktop publishing, graphic arts, color facsimile, newspaper wirephoto transmission, medical imaging, and many other continuous-tone image applications require a compression standard in order to develop significantly beyond their present state. JPEG has undertaken the ambitious task of developing a general-purpose compression standard to meet the needs of almost all continuous-tone still-image applications.

If this goal proves attainable, not only will individual applications flourish, but exchange of images across application boundaries will be facilitated. This latter feature will become increasingly important as more image applications are implemented on general-purpose computing systems, which are themselves becoming increasingly interoperable and internetworked. For applications which require specialized VLSI to meet their compression and decompression speed requirements, a common method will provide economies of scale not possible within a single application.

This article gives an overview of JPEG's proposed image-compression standard. Readers without prior knowledge of JPEG or compression based on the Discrete Cosine Transform (DCT) are encouraged to study first the detailed description of the Baseline sequential codec, which is the basis for all of the DCT-based decoders. While this article provides many details, many more are necessarily omitted. The reader should refer to the ISO draft standard [2] before attempting implementation.

Some of the earliest industry attention to the JPEG proposal has been focused on the Baseline sequential codec as a motion image compression method - of the "intraframe" class, where each frame is encoded as a separate image. This class of motion image coding, while providing less compression than "interframe" methods like MPEG, has greater flexibility for video editing. While this paper focuses only on JPEG as a still picture standard (as ISO intended), it is interesting to note that JPEG is likely to become a "de facto" intraframe motion standard as well.

2 Background: Requirements and Selection Process

JPEG's goal has been to develop a method for continuous-tone image compression which meets the following requirements:

1) be at or near the state of the art with regard to compression rate and accompanying image fidelity, over a wide range of image quality ratings, and especially in the range where visual fidelity to the original is characterized as "very good" to "excellent"; also, the encoder should be parameterizable, so that the application (or user) can set the desired compression/quality tradeoff;

2) be applicable to practically any kind of continuous-tone digital source image (i.e. for most practical purposes not be restricted to images of certain dimensions, color spaces, pixel aspect ratios, etc.) and not be limited to classes of imagery with restrictions on scene content, such as complexity, range of colors, or statistical properties;

3) have tractable computational complexity, to make feasible software implementations with viable performance on a range of CPU's, as well as hardware implementations with viable cost for applications requiring high performance;

4) have the following modes of operation:

   - Sequential encoding: each image component is encoded in a single left-to-right, top-to-bottom scan;
   - Progressive encoding: the image is encoded in multiple scans for applications in which transmission time is long, and the viewer prefers to watch the image build up in multiple coarse-to-clear passes;
   - Lossless encoding: the image is encoded to guarantee exact recovery of every source image sample value (even though the result is low compression compared to the lossy modes);
   - Hierarchical encoding: the image is encoded at multiple resolutions so that lower-resolution versions may be accessed without first having to decompress the image at its full resolution.

In June 1987, JPEG conducted a selection process based on a blind assessment of subjective picture quality, and narrowed 12 proposed methods to three. Three informal working groups formed to refine them, and in January 1988, a second, more rigorous selection process [19] revealed that the "ADCT" proposal [11], based on the 8x8 DCT, had produced the best picture quality.

At the time of its selection, the DCT-based method was only partially defined for some of the modes of operation. From 1988 through 1990, JPEG undertook the sizable task of defining, documenting, simulating, testing, validating, and simply agreeing on the plethora of details necessary for genuine interoperability and universality. Further history of the JPEG effort is contained in [6, 7, 9, 18].

3 Architecture of the Proposed Standard

The proposed standard contains the four "modes of operation" identified previously. For each mode, one or more distinct codecs are specified. Codecs within a mode differ according to the precision of source image samples they can handle or the entropy coding method they use. Although the word codec (encoder/decoder) is used frequently in this article, there is no requirement that implementations must include both an encoder and a decoder. Many applications will have systems or devices which require only one or the other.

The four modes of operation and their various codecs have resulted from JPEG's goal of being generic and from the diversity of image formats across applications. The multiple pieces can give the impression of undesirable complexity, but they should actually be regarded as a comprehensive "toolkit" which can span a wide range of continuous- tone image applications. It is unlikely that many implementations will utilize every tool -- indeed, most of the early implementations now on the market (even before final ISO approval) have implemented only the Baseline sequential codec.

The Baseline sequential codec is inherently a rich and sophisticated compression method which will be sufficient for many applications. Getting this minimum JPEG capability implemented properly and interoperably will provide the industry with an important initial capability for exchange of images across vendors and applications.

4 Processing Steps for DCT-Based Coding

Figures 1 and 2 show the key processing steps which are the heart of the DCT- based modes of operation. These figures illustrate the special case of single- component (grayscale) image compression. The reader can grasp the essentials of DCT- based compression by thinking of it as essentially compression of a stream of 8x8 blocks of grayscale image samples. Color image compression can then be approximately regarded as compression of multiple grayscale images, which are either compressed entirely one at a time, or are compressed by alternately interleaving 8x8 sample blocks from each in turn.

For DCT sequential- mode codecs, which include the Baseline sequential codec, the simplified diagrams indicate how single- component compression works in a fairly complete way. Each 8x8 block is input, makes its way through each processing step, and yields output in compressed form into the data stream. For DCT progressive- mode codecs, an image buffer exists prior to the entropy coding step, so that an image can be stored and then parceled out in multiple scans with successively improving quality. For the hierarchical mode of operation, the steps shown are used as building blocks within a larger framework.

4.1 8x8 FDCT and IDCT

At the input to the encoder, source image samples are grouped into 8x8 blocks, shifted from unsigned integers with range $[0, 2^P - 1]$ to signed integers with range $[-2^{P-1}, 2^{P-1}-1]$, and input to the Forward DCT (FDCT). At the output from the decoder, the Inverse DCT (IDCT) outputs 8x8 sample blocks to form the reconstructed image. The following equations are the idealized mathematical definitions of the 8x8 FDCT and 8x8 IDCT:

$$F(u,v) = \frac{1}{4} C(u)C(v)\left[\sum_{x=0}^{7} \sum_{y=0}^{7} f(x,y) * \cos\frac{(2x+1)u\pi}{16} \cos\frac{(2y+1)v\pi}{16} \right] \qquad (1)$$

$$f(x,y) = \frac{1}{4}\left[\sum_{u=0}^{7} \sum_{v=0}^{7} C(u)C(v)F(u,v) * \cos\frac{(2x+1)u\pi}{16} \cos\frac{(2y+1)v\pi}{16} \right] \qquad (2)$$

where: $C(u),\ C(v) = 1/\sqrt{2}$ for $u,v = 0$;

$C(u),\ C(v) = 1$ otherwise.

The DCT is related to the Discrete Fourier Transform (DFT). Some simple intuition for DCT- based compression can be obtained by viewing the FDCT as a harmonic analyzer and the IDCT as a harmonic synthesizer. Each 8x8 block of source image samples is effectively a 64- point discrete signal which is a function of the two spatial dimensions *x* and *y*. The FDCT takes such a signal as its input and decomposes it into 64 orthogonal basis signals. Each contains one of the 64 unique two- dimensional (2D) "spatial frequencies" which comprise the input signal's "spectrum." The ouput of the FDCT is the set of 64 basis- signal amplitudes or "DCT coefficients" whose values are uniquely determined by the particular 64- point input signal.

The DCT coefficient values can thus be regarded as the relative amount of the 2D spatial frequencies contained in the 64- point input signal. The coefficient with zero frequency in both dimensions is called the "DC coefficient" and the remaining 63 coefficients are called the "AC coefficients." Because sample values

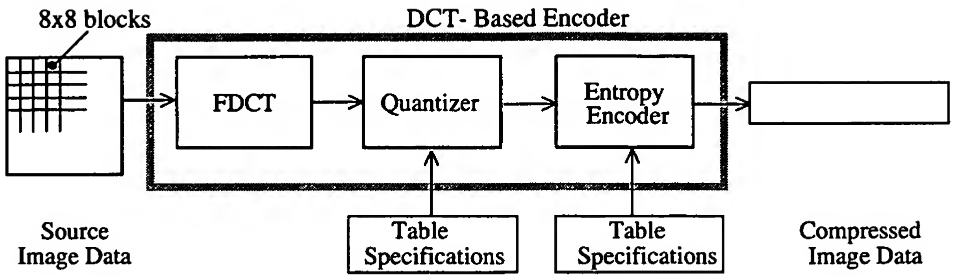


Figure 1. DCT- Based Encoder Processing Steps

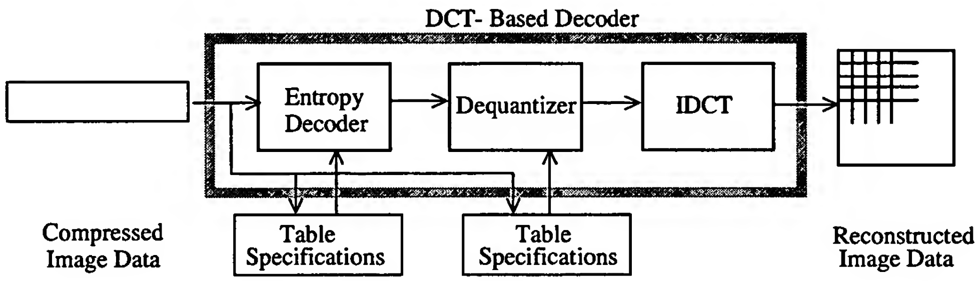


Figure 2. DCT- Based Decoder Processing Steps

typically vary slowly from point to point across an image, the FDCT processing step lays the foundation for achieving data compression by concentrating most of the signal in the lower spatial frequencies. For a typical 8x8 sample block from a typical source image, most of the spatial frequencies have zero or near- zero amplitude and need not be encoded.

At the decoder the IDCT reverses this processing step. It takes the 64 DCT coefficients (which at that point have been quantized) and reconstructs a 64- point ouput image signal by summing the basis signals. Mathematically, the DCT is one- to- one mapping for 64- point vectors between the image and the frequency domains. If the FDCT and IDCT could be computed with perfect accuracy and if the DCT coefficients were not quantized as in the following description, the original 64- point signal could be exactly recovered. In principle, the DCT introduces no loss to the source image samples; it merely transforms them to a domain in which they can be more efficiently encoded.

Some properties of practical FDCT and IDCT implementations raise the issue of what precisely should be required by the JPEG standard. A fundamental property is that the FDCT and IDCT equations contain transcendental functions. Consequently, no physical implementation can compute them with perfect accuracy. Because of the DCT's application importance and its relationship to the DFT, many different algorithms by which the FDCT and IDCT may be approximately computed have been devised [16]. Indeed, research in fast DCT algorithms is ongoing and no single algorithm is optimal for all implementations. What is optimal in software for a general- purpose CPU is unlikely to be optimal in firmware for a programmable DSP and is certain to be suboptimal for dedicated VLSI.

Even in light of the finite precision of the DCT inputs and outputs, independently designed implementations of the very same FDCT or IDCT algorithm which differ even minutely in the precision by which they represent cosine terms or intermediate results, or in the way they sum and round fractional values, will eventually produce slightly different outputs from identical inputs.

To preserve freedom for innovation and customization within implementations, JPEG has chosen to specify neither a unique FDCT algorithm or a unique IDCT algorithm in its proposed standard. This makes compliance somewhat more difficult to confirm, because two compliant encoders (or decoders) generally will not produce identical outputs given identical inputs. The JPEG standard will address this issue by specifying an accuracy test as part of its compliance tests for all DCT- based encoders and decoders; this is to ensure against crudely inaccurate cosine basis functions which would degrade image quality.

For each DCT-based mode of operation, the JPEG proposal specifies separate codecs for images with 8-bit and 12-bit (per component) source image samples. The 12-bit codecs, needed to accommodate certain types of medical and other images, require greater computational resources to achieve the required FDCT or IDCT accuracy. Images with other sample precisions can usually be accommodated by either an 8-bit or 12-bit codec, but this must be done outside the JPEG standard. For example, it would be the responsibility of an application to decide how to fit or pad a 6-bit sample into the 8-bit encoder's input interface, how to unpack it at the decoder's output, and how to encode any necessary related information.

4.2 Quantization

After output from the FDCT, each of the 64 DCT coefficients is uniformly quantized in conjunction with a 64-element Quantization Table, which must be specified by the application (or user) as an input to the encoder. Each element can be any integer value from 1 to 255, which specifies the step size of the quantizer for its corresponding DCT coefficient. The purpose of quantization is to achieve further compression by representing DCT coefficients with no greater precision than is necessary to achieve the desired image quality. Stated another way, the goal of this processing step is to discard information which is not visually significant. Quantization is a many-to-one mapping, and therefore is fundamentally lossy. It is the principal source of lossiness in DCT-based encoders.

Quantization is defined as division of each DCT coefficient by its corresponding quantizer step size, followed by rounding to the nearest integer:

$$F^Q(u,v) \;=\; Integer\,Round\,\left(\frac{F(u,v)}{Q(u,v)}\right) \qquad (3)$$

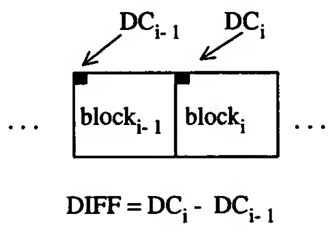
This output value is normalized by the quantizer step size. Dequantization is the inverse function, which in this case means simply that the normalization is removed by multiplying by the step size, which returns the result to a representation appropriate for input to the IDCT:

$$F^{Q'}(u,v) \;=\; F^Q(u,v) \,*\, Q(u,v) \qquad (4)$$

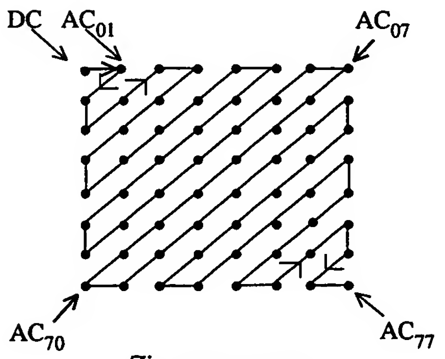
When the aim is to compress the image as much as possible without visible artifacts, each step size ideally should be chosen as the perceptual threshold or 'just noticeable difference" for the visual contribution of its corresponding cosine basis function. These thresholds are also functions of the source image characteristics, display characteristics and viewing distance. For applications in which these variables can be reasonably well defined, psychovisual experiments can be performed to determine the best thresholds. The experiment described in [12] has led to a set of Quantization Tables for CCIR-601 [4] images and displays. These have been used experimentally by JPEG members and will appear in the ISO standard as a matter of information, but not as a requirement.

4.3 DC Coding and Zig-Zag Sequence

After quantization, the DC coefficient is treated separately from the 63 AC coefficients. The DC coefficient is a measure of the average value of the 64 image samples. Because there is usually strong correlation between the DC coefficients of adjacent 8x8 blocks, the quantized DC coefficient is encoded as the difference from the DC term of the previous block in the encoding order (defined in the following), as shown in Figure 3. This special treatment is worthwhile, as DC coefficients frequently contain a significant fraction of the total image energy.



Differential DC encoding



Zig–zag sequence

Figure 3. Preparation of Quantized Coefficients for Entropy Coding

Finally, all of the quantized coefficients are ordered into the "zig-zag" sequence, also shown in Figure 3. This ordering helps to facilitate entropy coding by placing low-frequency coefficients (which are more likely to be nonzero) before high-frequency coefficients.

4.4 Entropy Coding

The final DCT-based encoder processing step is entropy coding. This step achieves additional compression losslessly by encoding the quantized DCT coefficients more compactly based on their statistical characteristics. The JPEG proposal specifies two entropy coding methods - Huffman coding [8] and arithmetic coding [15]. The Baseline sequential codec uses Huffman coding, but codecs with both methods are specified for all modes of operation.

It is useful to consider entropy coding as a 2-step process. The first step converts the zig-zag sequence of quantized coefficients into an intermediate sequence of symbols. The second step converts the symbols to a data stream in which the symbols no longer have externally identifiable boundaries. The form and definition of the intermediate symbols is dependent on both the DCT-based mode of operation and the entropy coding method.

Huffman coding requires that one or more sets of Huffman code tables be specified by the application. The same tables used to compress an image are needed to decompress it. Huffman tables may be predefined and used within an application as defaults, or computed specifically for a given image in an initial statistics-gathering pass prior to compression. Such choices are the business of the applications which use JPEG; the JPEG proposal specifies no required Huffman tables. Huffman coding for the Baseline sequential encoder is described in detail in section 7.

By contrast, the particular arithmetic coding method specified in the JPEG proposal [2] requires no tables to be externally input, because it is able to adapt to the image statistics as it encodes the image. (If desired, statistical conditioning tables can be used as inputs for slightly better efficiency, but this is not required.) Arithmetic coding has produced 5-10% better compression than Huffman for many of the images which JPEG members have tested. However, some feel it is more complex than Huffman coding for certain implementations, for example, the highest-speed hardware implementations. (Throughout JPEG's history, "complexity" has proved to be most elusive as a practical metric for comparing compression methods.)

If the only difference between two JPEG codecs is the entropy coding method, transcoding between the two is possible by simply entropy decoding with one method and entropy recoding with the other.

4.5 Compression and Picture Quality

For color images with moderately complex scenes, all DCT-based modes of operation typically produce the following levels of picture quality for the indicated ranges of compression. These levels are only a guideline - quality and compression can vary significantly according to source image characteristics and scene content. (The units "bits/pixel" here mean the total number of bits in the compressed image - including the chrominance components - divided by the number of samples in the luminance component.)

- 0.25-0.5 bits/pixel: moderate to good quality, sufficient for some applications;
- 0.5-0.75 bits/pixel: good to very good quality, sufficient for many applications;
- 0.75-1/5 bits/pixel: excellent quality, sufficient for most applications;
- 1.5-2.0 bits/pixel: usually indistinguishable from the original, sufficient for the most demanding applications.

5 Processing Steps for Predictive Lossless Coding

After its selection of a DCT-based method in 1988, JPEG discovered that a DCT-based lossless mode was difficult to define as a practical standard against which encoders and decoders could be independently implemented, without placing severe constraints on both encoder and decoder implementations.

JPEG, to meet its requirement for a lossless mode of operation, has chosen a simple predictive method which is wholly independent of the DCT processing described previously. Selection of this method was not the result of rigorous competitive evaluation as was the DCT-based method. Nevertheless, the JPEG lossless method produces results which, in light of its simplicity, are surprisingly close to the state of the art for lossless continuous-tone compression, as indicated by a recent technical report [5].

Figure 4 shows the main processing steps for a single-component image. A predictor combines the values of up to three neighboring samples (A, B, and C) to form a prediction of the sample indicated by X in Figure 5. This prediction is then subtracted from the actual value of sample X, and the difference is encoded

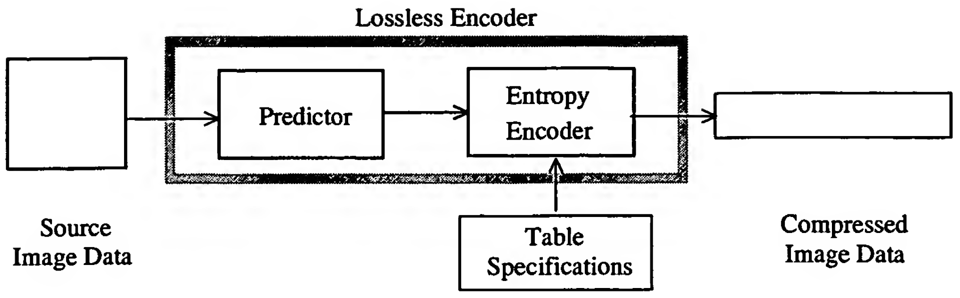


Figure 4. Lossless Mode Encoder Processing Steps

losslessly by either of the entropy coding methods - Huffman or arithmetic. Any one of the eight predictors listed in Table 1 (under 'selection- value') can be used.

Selections 1, 2, and 3 are one- dimensional predictors and selections 4, 5, 6 and 7 are two- dimensional predictors. Selection- value 0 can only be used for differential coding in the hierarchical mode of operation. The entropy coding is nearly identical to that used for the DC coefficient as described in section 7.1 (for Huffman coding).

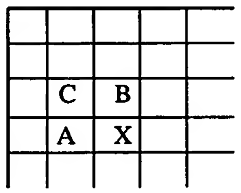


Figure 5. 3- Sample Prediction Neighborhood

For the lossless mode of operation, two different codecs are specified - one for each entropy coding method. The encoders can use any source image precision from 2 to 16 bits/sample, and can use any of the predictors except selection- value 0. The decoders must handle any of the sample precisions and any of the predictors. Lossless codecs typically produce around 2:1 compression for color images with moderately complex scenes.

selection-value	prediction
0	no prediction
1	A
2	B
3	C
4	A+B- C
5	A+((B- C)/2)
6	B+((A- C)/2)
7	(A+B)/2

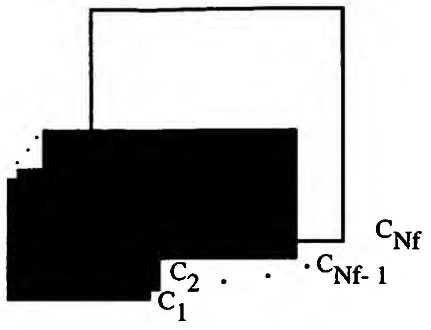
Table 1. Predictors for Lossless Coding

6 Multiple-Component Images

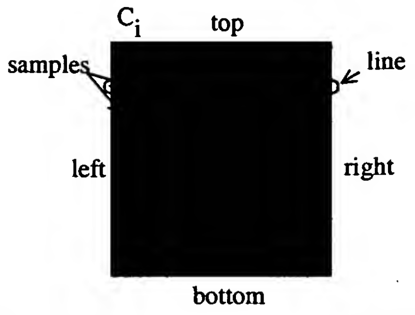
The previous sections discussed the key processing steps of the DCT- based and predictive lossless codecs for the case of single- component source images. These steps accomplish the image data compression. But a good deal of the JPEG proposal is also concerned with the handling and control of color (or other) images with multiple components. JPEG's aim for a generic compression standard requires its proposal to accommodate a variety of source image formats.

6.1 Source Image Formats

The source image model used in the JPEG proposal is an abstraction from a variety of image types and applications and consists of only what is necessary to compress and reconstruct digital image data. The reader should recognize that the JPEG compressed data format does not encode enough information to serve as a complete image representation. For example, JPEG does not specify or encode any information on pixel aspect ratio, color space, or image acquisition characteristics.



(a) Source image with multiple components



(b) Characteristics of an image component

Figure 6. JPEG Source Image Model

Figure 6 illustrates the JPEG source image model. A source image contains from 1 to 255 image components, sometimes called color or spectral bands or channels. Each component consists of a rectangular array of samples. A sample is defined to be an unsigned integer with precision P bits, with any value in the range [0, 2^P- 1]. All samples of all components within the same source image must have the same precision P. P can be 8 or 12 for DCT- based codecs, and 2 to 16 for predictive codecs.

The ith component has sample dimensions x_i by y_i. To accommodate formats in which some image components are sampled at different rates than others, components can have different dimensions. The dimensions must have a mutual integral relationship defined by H_i and V_i, the relative horizontal and vertical sampling factors, which must be specified for each component. Overall image dimensions X and Y are defined as the maximum x_i and y_i for all components in the image, and can be any number up to 2^{16}. H and V are allowed only the integer values 1 through 4. The encoded parameters are X, Y, and H_is and V_is for each components. The decoder reconstructs the dimensions x_i and y_i for each component, according to the following relationship shown in Equation 5:

$$x_i = \left\lceil X \times \frac{H_i}{H_{max}} \right\rceil \text{ and}$$

$$y_i = \left\lceil Y \times \frac{V_i}{V_{max}} \right\rceil \qquad (5)$$

where $\lceil\ \rceil$ is the ceiling function.

6.2 Encoding Order and Interleaving

A practical image compression standard must address how systems will need to handle the data during the process of decompression. Many applications need to pipeline the process of displaying or printing multiple- component images in parallel with the process of decompression. For many systems, this is only feasible if the components are interleaved together within the compressed data stream.

To make the same interleaving machinery applicable to both DCT- based and predictive codecs, the JPEG proposal has defined the concept of "data unit." A data unit is a sample in predictive codecs and an 8x8 block of samples in DCT- based codecs.

The order in which compressed data units are placed in the compressed data stream is a generalization of raster- scan order. Generally, data units are ordered from left- to- right and top- to- bottom according to the orientation shown in Figure 6. (It is the responsibility of applications to define which edges of a source image are top, bottom, left and right.) If an image component is noninterleaved (i.e., compressed without being interleaved with other components), compressed data units are ordered in a pure raster scan as shown in Figure 7.

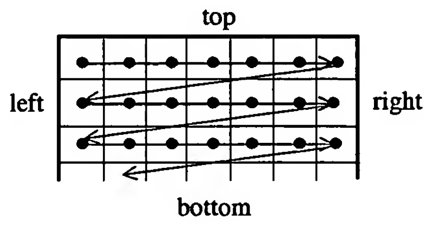
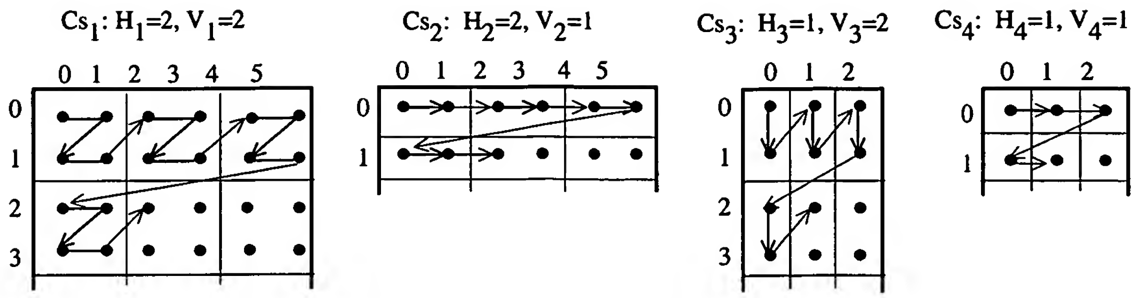


Figure 7. Noninterleaved Data Ordering

When two or more components are interleaved, each component C_i is partitioned into rectangular regions of H_i by V_i data units, as shown in the generalized example of Figure 8. Regions are ordered within a component from left- to- right and top- to- bottom, and within a region, data units are ordered from left- to- right and top- to- bottom. The JPEG proposal defines the term Minimum Coded Unit (MCU) to be the smallest



$$\begin{aligned}
MCU_1 &= d^1_{00}\, d^1_{01}\, d^1_{10}\, d^1_{11} \;\; d^2_{00}\, d^2_{01} \;\; d^3_{00}\, d^3_{10} \;\; d^4_{00},\\
MCU_2 &= d^1_{02}\, d^1_{03}\, d^1_{12}\, d^1_{13} \;\; d^2_{02}\, d^2_{03} \;\; d^3_{01}\, d^3_{11} \;\; d^4_{01},\\
MCU_3 &= d^1_{04}\, d^1_{05}\, d^1_{14}\, d^1_{15} \;\; d^2_{04}\, d^2_{05} \;\; d^3_{02}\, d^3_{12} \;\; d^4_{02},\\
MCU_4 &= \underbrace{d^1_{20}\, d^1_{21}\, d^1_{30}\, d^1_{31}}_{Cs_1 \text{ data units}} \;\; \underbrace{d^2_{10}\, d^2_{11}}_{Cs_2} \;\; \underbrace{d^3_{20}\, d^3_{30}}_{Cs_3} \;\; \underbrace{d^4_{10},}_{Cs_4}
\end{aligned}$$

Figure 8. Generalized Interleaved Data Ordering Example

group of interleaved data units. For the example shown, MCU_1 consists of data units taken first from the top- left- most region of C_1, followed by data units from the same region of C_2, and likewise for C_3 and C_4. MCU_2 continues the pattern as shown.

Thus, interleaved data is an ordered sequence of MCUs, and the number of data units contained in an MCU is determined by the number of components interleaved and their relative sampling factors. The maximum number of components which can be interleaved is 4 and the maximum number of data units in an MCU is 10. The latter restriction is expressed as shown in Equation 6, where the summation is over the interleaved components:

$$\sum_{\substack{\textit{all}\ \text{i in}\\ \text{interleave}}} H_i \times V_i \le 10 \qquad (6)$$

Because of this restriction, not every combination of 4 components which can be represented in noninterleaved order within a JPEG- compressed image is allowed to be interleaved. Also, note that the JPEG proposal allows some components to be interleaved and some to be noninterleaved within the same compressed image.

6.3 Multiple Tables

In addition to the interleaving control discussed previously, JPEG codecs must control application of the proper table data to the proper components. The same quantization table and the same entropy coding table (or set of tables) must be used to encode all samples within a component.

JPEG decoders can store up to 4 different quantization tables and up to 4 different (sets of) entropy coding tables simultaneously. (The Baseline sequential decoder is the exception; it can only store up to 2 sets of entropy coding tables.) This is necessary for switching between different tables during decompression of a scan containing multiple (interleaved) components, in order to apply the proper table to the proper component. (Tables cannot be loaded during decompression of a scan.) Figure 9 illustrates the table- switching control that must be managed in conjunction with multiple- component interleaving for the encoder side. (This simplified view does not distinguish between quantization and entropy coding tables.)

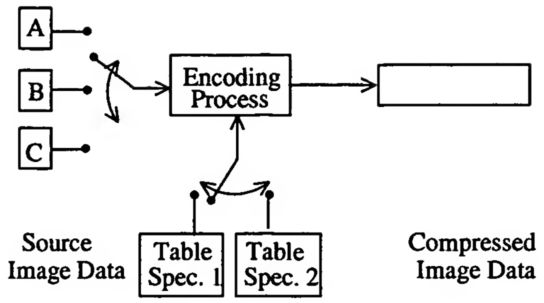


Figure 9. Component- Interleave and Table- Switching Control

7 Baseline and Other DCT Sequential Codecs

The DCT sequential mode of operation consists of the FDCT and Quantization steps from section 4, and the multiple- component control from section 6.3. In addition to the Baseline sequential codec, other DCT sequential codecs are defined to accommodate the two different sample precisions (8 and 12 bits) and the two different types of entropy coding methods (Huffman and arithmetic).

Baseline sequential coding is for images with 8- bit samples and uses Huffman coding only. It also differs from the other sequential DCT codecs in that its decoder can store only two sets of Huffman tables (one AC table and DC table per set). This restriction means that, for images with three or four interleaved components, at least one set of Huffman tables must be shared by two components. This restriction poses no limitation at all for noninterleaved components; a new set of tables can be loaded into the decoder before decompression of a noninterleaved component begins.

For many applications which do need to interleave three color components, this restriction is hardly a limitation at all. Color spaces (YUV, CIELUV, CIELAB, and others) which represent the chromatic (''color'') information in two components and the achromatic (''grayscale'') information in a third are more efficient for compression than spaces like RGB. One Huffman table set can be used for the achromatic component and one for the chrominance components. DCT coefficient statistics are similar for the chrominance components of most images, and one set of Huffman tables can encode both almost as optimally as two.

The committee also felt that early availability of single- chip implementations at commodity prices would encourage early acceptance of the JPEG proposal in a variety of applications. In 1988 when Baseline sequential was defined, the committee's VLSI experts felt that current technology made the feasibility of crowding four sets of loadable Huffman tables - in addition to four sets of Quantization tables - onto a single commodity- priced codec chip a risky proposition.

The FDCT, Quantization, DC differencing, and zig- zag ordering processing steps for the Baseline sequential codec proceed just as described in section 4. Prior to entropy coding, there usually are few nonzero and many zero- valued coefficients. The task of entropy coding is to encode these few coefficients efficiently. The description of Baseline sequential entropy coding is given in two steps: conversion of the quantized DCT coefficients into an intermediate sequence of symbols and assignment of variable- length codes to the symbols.

7.1 Intermediate Entropy Coding Representations

In the intermediate symbol sequence, each nonzero AC coefficient is represented in combination with the ''runlength'' (consecutive number) of zero- valued AC coefficients which precede it in the zig- zag sequence. Each such runlength/nonzero- coefficient combination is (usually) represented by a pair of symbols:

symbol- 1	symbol- 2
(RUNLENGTH, SIZE)	(AMPLITUDE)

Symbol- 1 represents two pieces of information, RUNLENGTH and SIZE. Symbol- 2 represents the single piece of information designated AMPLITUDE, which is simply the amplitude of the nonzero AC coefficient. RUNLENGTH is the number of consecutive zero- valued AC coefficients in the zig- zag sequence preceding the nonzero AC coefficient being represented. SIZE is the number of bits used to encode AMPLITUDE - that is, to encoded symbol- 2, by the signed- integer encoding used with JPEG's particular method of Huffman coding.

RUNLENGTH represents zero- runs of length 0 to 15. Actual zero- runs in the zig- zag sequence can be greater than 15, so the symbol- 1 value (15, 0) is interpreted as the extension symbol with runlength=16. There can be up to three consecutive (15, 0) extensions before the terminating symbol- 1 whose RUNLENGTH value completes the actual runlength. The terminating symbol- 1 is always followed by a single symbol- 2, except for the case in which the last run of zeros includes the last (63d) AC coefficient. In this frequent case, the special symbol- 1 value (0,0) means EOB (end of block), and can be viewed as an ''escape'' symbol which terminates the 8x8 sample block.

Thus, for each 8x8 block of samples, the zig-zag sequence of 63 quantized AC coefficients is represented as a sequence of symbol-1, symbol-2 symbol pairs, though each "pair" can have repetitions of symbol-1 in the case of a long run-length or only one symbol-1 in the case of an EOB.

The possible range of quantized AC coefficients determines the range of values which both the AMPLITUDE and the SIZE information must represent. A numerical analysis of the 8x8 FDCT equation shows that, if the 64-point (8x8 block) input signal contains N-bit integers, then the nonfractional part of the output numbers (DCT coefficients) can grow by at most 3 bits. This is also the largest possible size of a quantized DCT coefficient when its quantizer step size has integer value 1.

Baseline sequential has 8-bit integer source samples in the range $[-2^7, 2^7-1]$, so quantized AC coefficient amplitudes are covered by integers in the range $[-2^{10}, 2^{10}-1]$. The signed-integer encoding uses symbol-2 AMPLITUDE codes of 1 to 10 bits in length (so SIZE also represents values from 1 to 10), and RUNLENGTH represents values from 0 to 15 as discussed previously. For AC coefficients, the structure of the symbol-1 and symbol-2 intermediate representations is illustrated in Tables 2 and 3, respectively.

The intermediate representation for an 8x8 sample block's differential DC coefficient is structured similarly. Symbol-1, however, represents only SIZE information; symbol-2 represents AMPLITUDE information as before:

symbol-1	symbol-2
(SIZE)	(AMPLITUDE)

Because the DC coefficient is differentially encoded, it is covered by twice as many integer values, $[-2^{11}, 2^{11}-1]$ as the AC coefficients, so one additional level must be added to the bottom of Table 3 for DC coefficients. Symbol-1 for DC coefficients thus represents a value from 1 to 11.

		SIZE					
		0	1	2	...	9	10
RUN LENGTH	0	EOB					
	.	X	RUN-SIZE values				
	.	X					
	.	X					
	15	ZRL					

Table 2. Baseline Huffman Coding Symbol-1 Structure

7.2 Variable-Length Entropy Coding

Once the quantized coefficient data for an 8x8 block is represented in the intermediate symbol sequence described above, variable-length codes are assigned. For each 8x8 block, the DC coefficient's symbol-1 and symbol-2 representation is coded and output first.

For both DC and AC coefficients, each symbol-1 is encoded with a variable-length code (VLC) from the Huffman table set assigned to the 8x8 block's image component. Each symbol-2 is encoded with a "variable-length integer" (VLI) code whose length in bits is given in Table 3. VLCs and VLIs both are codes with variable lengths, but VLIs are not Huffman codes. An important distinction is that the length of a VLC (Huffman code) is not known until it is decoded, but the length of a VLI is stored in its preceding VLC.

Huffman codes (VLCs) must be specified externally as an input to JPEG encoders. (Note that the form in which Huffman tables are represented in the data stream is an indirect specification with which the decoder must construct the tables themselves prior to decompression.) The JPEG proposal includes an example set of Huffman tables in its information annex, but because they are application-specific, it specifies none for required use. The VLI codes in contrast, are "hardwired" into the proposal. This is appropriate, because the VLI codes are far more numerous, can be computed rather than stored, and have not been shown to be appreciably more efficient when implemented as Huffman codes.

7.3 Baseline Encoding Example

This section gives an example of Baseline compression and encoding of a single 8x8 sample block. Note that a good deal of the operation of a complete JPEG Baseline encoder is omitted here, including creation of Interchange Format information (parameters, headers, quantization and Huffman tables), byte-stuffing, padding to byte-boundaries prior to a marker code, and other key operations. Nonetheless, this example should help to make concrete much of the foregoing explanation.

Figure 10(a) is an 8x8 block of 8-bit samples, aribtrarily extracted from a real image. The small variations from sample to sample indicate the predominance of low spatial frequencies. After subtracting 128 from each sample for the required level-shift, the 8x8 block is input to the FDCT, equation (1). Figure 10(b) shows (to one decimal place) the resulting DCT coefficients. Except for a few of the lowest frequency coefficients, the amplitudes are quite small.

139	144	149	153	155	155	155	155
144	151	153	156	159	156	156	156
150	155	160	163	158	156	156	156
159	161	162	160	160	159	159	159
159	160	161	162	162	155	155	155
161	161	161	161	160	157	157	157
162	162	161	163	162	157	157	157
162	162	161	161	163	158	158	158

(a) source image samples

235.6	- 1.0	- 12.1	- 5.2	2.1	- 1.7	- 2.7	1.3
- 22.6	- 17.5	- 6.2	- 3.2	- 2.9	- 0.1	0.4	- 1.2
- 10.9	- 9.3	- 1.6	1.5	0.2	- 0.9	- 0.6	- 0.1
- 7.1	- 1.9	0.2	1.5	0.9	- 0.1	0.0	0.3
- 0.6	- 0.8	1.5	1.6	- 0.1	- 0.7	0.6	1.3
1.8	- 0.2	1.6	- 0.3	- 0.8	1.5	1.0	- 1.0
- 1.3	- 0.4	- 0.3	- 1.5	- 0.5	1.7	1.1	- 0.8
- 2.6	1.6	- 3.8	- 1.8	1.9	1.2	- 0.6	- 0.4

(b) forward DCT coefficients

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

(c) quantization table

15	0	- 1	0	0	0	0	0
- 2	- 1	0	0	0	0	0	0
- 1	- 1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

(d) normalized quantized coefficients

240	0	- 10	0	0	0	0	0
- 24	- 12	0	0	0	0	0	0
- 14	- 13	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

(e) denormalized quantized coefficients

144	146	149	152	154	156	156	156
148	150	152	154	156	156	156	156
155	156	157	158	158	157	156	155
160	161	161	162	161	159	157	155
163	163	164	163	162	160	158	156
163	164	164	164	162	160	158	157
160	161	162	162	162	161	159	158
158	159	161	161	162	161	159	158

(f) reconstructed image samples

Figure 10. DCT and Quantization Examples

Figure 10(c) is the example quantization table for luminance (grayscale) components included in the informational annex of the draft JPEG standard part 1 [2]. Figure 10(d) shows the quantized DCT coefficients, normalized by their quantization table entries, as specified by equation (3). At the decoder these numbers are "denormalized" according to equation (4), and input to the IDCT, equation (2). Finally, figure 10(f) shows the reconstructed sample values, remarkably similar to the originals in 10(a).

Of course, the numbers in figure 10(d) must be Huffman- encoded before transmission to the decoder. The first number of the block to be encoded is the DC term, which must be differentially encoded. If the quantized DC term of the previous block is, for example, 12, then the difference is +3. Thus, the intermediate representation is (2)(3), for SIZE=2 and AMPLITUDE=3.

Next, the the quantized AC coefficients are encoded. Following the zig- zag order, the first non- zero coefficient is - 2, preceded by a zero- run of 1. This yields an intermediate representation of (1,2)(- 2). Next encountered in the zig- zag order are three consecutive non- zeros of amplitude - 1. This means each is preceded by a zero- run of length zero, for intermediate symbols (0,1)(- 1). The last non- zero coefficient is - 1 preceded by two zeros, for (2,1)(- 1). Because this is the last non- zero coefficient, the final symbol representing this 8x8 block is EOB, or (0,0).

Thus, the intermediate sequence of symbols for this example 8x8 block is:

(2)(3), (1,2)(- 2), (0,1)(- 1), (0,1)(- 1),
(0,1)(- 1), (2,1)(- 1), (0,0)

Next the codes themselves must be assigned. For this example, the VLCs (Huffman codes) from the informational annex of [2] will be used. The differential- DC VLC for this example is:

(2)	011

The AC luminance VLCs for this example are:

(0,0)	1010
(0,1)	00
(1,2)	11011
(2,1)	11100

The VLIs specified in [2] are related to the two's complement representation. They are:

(3)	11
(- 2)	01
(- 1)	0

Thus, the bit- stream for this 8x8 example block is as follows. Note that 31 bits are required to represent 64 coefficients, which achieves compression of just under 0.5 bits/sample:

0111111011010000000001110001010

7.4 Other DCT Sequential Codecs

The structure of the 12- bit DCT sequential codec with Huffman coding is a straightforward extension of the entropy coding method described previously. Quantized DCT coefficients can be 4 bits larger, so the SIZE and AMPLITUDE information extend accordingly. DCT sequential with arithmetic coding is described in detail in [2].

8 DCT Progressive Mode

The DCT progressive mode of operation consists of the same FDCT and Quantization steps (from section 4) that are used by DCT sequential mode. The key difference is that each image component is encoded in multiple scans rather than in a single scan. The first scan(s) encode a rough but recognizable version of the image which can be transmitted quickly in comparison to the total transmission time, and are refined by succeeding scans until reaching a level of picture quality that was established by the quantization tables.

To achieve this requires the addition of an image- sized buffer memory at the output of the quantizer, before the input to entropy encoder. The buffer memory must be of sufficient size to store the image as quantized DCT coefficients, each of which (if stored straightforwardly) is 3 bits larger than the source image samples. After each block of DCT coefficients is quantized, it is stored in the coefficient buffer memory. The buffered coefficients are then partially encoded in each of multiple scans.

There are two complementary methods by which a block of quantized DCT coefficients may be partially encoded. First, only a specified 'band" of coefficients from the zig- zag sequence need be encoded within a given scan. This procedure is called 'spectral selection," because each band typically contains coefficients which occupy a lower or higher part of the spatial- frequency spectrum for that 8x8 block. Secondly, the coefficients within the current band need not be encoded to their full (quantized) accuracy in a given scan. Upon a coefficient's first encoding, the N most significant bits can be encoded first, where N is specifiable. In subsequent scans, the less significant bits can then be encoded. This procedure is called ''successive approximation.'' Both procedures can be used separately, or mixed in flexible combinations.

SIZE	AMPLITUDE
1	- 1,1
2	- 3,- 2,2,3
3	- 7..- 4,4..7
4	- 15..- 8,8..15
5	- 31..- 16,16..31
6	- 63..- 32,32..63
7	- 127..- 64,64..127
8	- 255..- 128,128..255
9	- 511..- 256,256..511
10	- 1023..- 512,512..1023

Table 3. Baseline Entropy Coding Symbol- 2 Structure

Some intuition for spectral selection and successive approximation can be obtained from Figure 11. The quantized DCT coefficient information can be viewed as a rectangle for which the axes are the DCT coefficients (in zig- zag order) and their amplitudes. Spectral selection slices the information in one dimension and successive approximation in the other.

9 Hierarchical Mode of Operation

The hierarchical mode provides a 'pyramidal" encoding of an image at multiple resolutions, each differing in resolution from its adjacent encoding by a factor of two in either the horizontal or vertical dimension or both. The encoding procedure can be summarized as follows:

1) Filter and down- sample the original image by the desired number of multiples of 2 in each dimension.

2) Encode this reduced- size image using one of the sequential DCT, progressive DCT, or lossless encoders described previously.

3) Decode this reduced- size image and then interpolate and up- sample it by 2 horizontally and/or vertically, using the identical interpolation filter which the receiver must use.

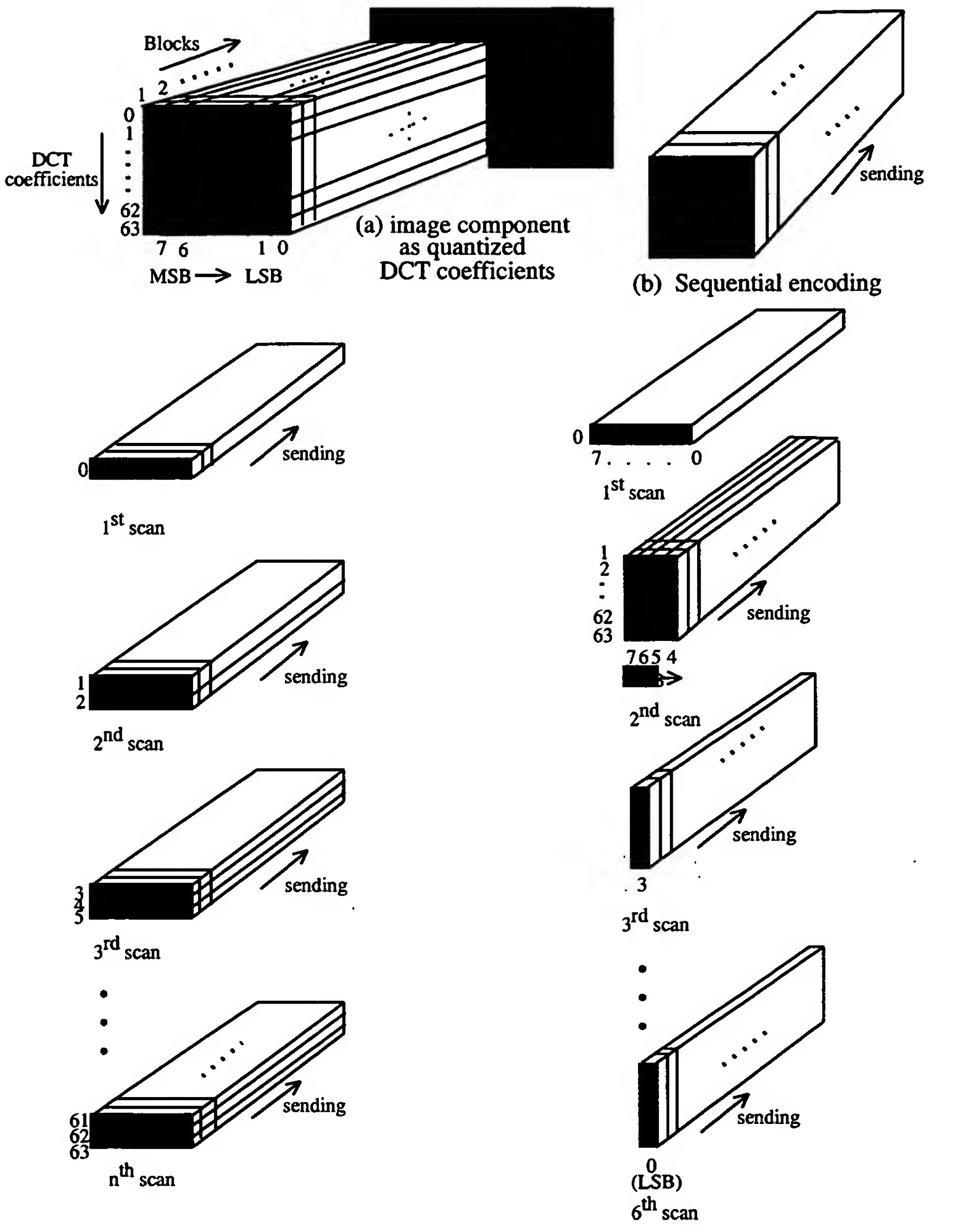


Figure 11. Spectral Selection and Successive Approximation Methods of Progressive Encoding

4) Use this up- sampled image as a prediction of the original at this resolution, and encode the difference image using one of the sequential DCT, progressive DCT, or lossless encoders described previously.

5) Repeat steps 3) and 4) until the full resolution of the image has been encoded.

The encoding in steps 2) and 4) must be done using only DCT- based processes, only lossless processes, or DCT- based processes with a final lossless process for each component.

Hierarchical encoding is useful in applications in which a very high resolution image must be accessed by a lower- resolution display. An example is an image scanned and compressed at high resolution for a very high- quality printer, where the image must also be displayed on a low- resolution PC video screen.

10 Other Aspects of the JPEG Proposal

Some key aspects of the proposed standard can only be mentioned briefly. Foremost among these are points concerning the coded representation for compressed image data specified in addition to the encoding and decoding procedures.

Most importantly, an *interchange format* syntax is specified which ensures that a JPEG-compressed image can be exchanged successfully between different application environments. The format is structured in a consistent way for all modes of operation. The interchange format always includes all quantization and entropy-coding tables which were used to compress the image.

Applications (and application-specific standards) are the "users" of the JPEG standard. The JPEG standard imposes no requirement that, within an application's environment, all or even any tables must be encoded with the compressed image data during storage or transmission. This leaves applications the freedom to specify default or referenced tables if they are considered appropriate. It also leaves them the responsibility to ensure that JPEG-compliant decoders used within their environment get loaded with the proper tables at the proper times, and that the proper tables are included in the interchange format when a compressed image is "exported" outside the application.

Some of the important applications that are already in the process of adopting JPEG compression or have stated their interest in doing so are Adobe's PostScript language for printing systems [1], the Raster Content portion of the ISO Office Document Architecture and Interchange Format [13], the future CCITT color facsimile standard, and the European ETSI videotext standard [10].

11 Standardization Schedule

JPEG's ISO standard will be divided into two parts. Part 1 [2] will specify the four modes of operation, the different codecs specified for those modes, and the interchange format. It will also contain a substantial informational section on implementation guidelines. Part 2 [3] will specify the compliance tests which will determine whether an encoder implementation, a decoder implementation, or a JPEG-compressed image in interchange format comply with the Part 1 specifications. In addition to the ISO documents referenced, the JPEG standard will also be issued as CCITT Recommendation T.81.

There are two key balloting phases in the ISO standardization process: a Committee Draft (CD) is balloted to determine promotion to Draft International Standard (DIS), and a DIS is balloted to determine promotion to International Standard (IS). A CD ballot requires four to six months of processing, and a DIS ballot requires six to nine months of processing. JPEG's Part 1 began DIS ballot in November 1991, and Part 2 began CD ballot in December 1991.

Though there is no guarantee that the first ballot of each phase will result in promotion to the next, JPEG achieved promotion of CD Part 1 to DIS Part 1 in the first ballot. Moreover, JPEG's DIS Part 1 has undergone no technical changes (other than some minor corrections) since JPEG's final Working Draft (WD) [14]. Thus, Part 1 has remained unchanged from the final WD, through CD, and into DIS. If all goes well, Part 1 should receive final approval as an IS in mid-1992, with Part 2 getting final IS approval about nine months later.

12 Conclusions

The emerging JPEG continuous-tone image compression standard is not a panacea that will solve the myriad issues which must be addressed before digital images will be fully integrated within all the applications that will ultimately benefit from them. For example, if two applications cannot exchange uncompressed images because they use incompatible color spaces, aspect ratios, dimensions, etc. then a common compression method will not help.

However, a great many applications are "stuck" because of storage or transmission costs, because of argument over which (nonstandard) compression method to use, or because VLSI codecs are too expensive due to low volumes. For these applications, the thorough technical evaluation, testing, selection, validation, and documentation work which JPEG committee members have performed is expected to soon yield an approved international standard that will withstand the tests of quality and time. As diverse imaging applications become increasingly implemented on open networked computing systems, the ultimate measure of the committee's success will be when JPEG-compressed digital images come to be regarded and even taken for granted as "just another data type," as text and graphics are today.

For more information

Information on how to obtain the ISO JPEG (draft) standards can be obtained by writing the author at the following address:

Digital Equipment Corporation
146 Main Street, ML01- 2/U44
Maynard, MA 01754- 2571

Internet: wallace@gauss.enet.dec.com

Floppy disks containing uncompressed, compressed, and reconstructed data for the purpose of informally validating whether an encoder or decoder implementation conforms to the proposed standard are available. Thanks to the following JPEG committee member and his company who have agreed to provide these for a nominal fee on behalf of the committee until arrangements can be made for ISO to provide them:

Eric Hamilton
C- Cube Microsystems
1778 McCarthy Blvd.
Milpitas, CA 95035

Acknowledgments

The following longtime JPEG core members have spent untold hours (usually in addition to their "real jobs") to make this collaborative international effort succeed. Each has made specific substantive contributions to the JPEG proposal: Aharon Gill (Zoran, Israel), Eric Hamilton (C- Cube, USA), Alain Leger (CCETT, France), Adriaan Ligtenberg (Storm, USA), Herbert Lohscheller (ANT, Germany), Joan Mitchell (IBM, USA), Michael Nier (Kodak, USA), Takao Omachi (NEC, Japan), William Pennebaker (IBM, USA), Henning Poulsen (KTAS, Denmark), and Jorgen Vaaben (AutoGraph, Denmark). The leadership efforts of Hiroshi Yasuda (NTT, Japan), the Convenor of JTC1/SC2/WG8 from which JPEG was spawned, Istvan Sebestyen (Siemens, Germany), the Special Rapporteur from CCITT SGVIII, and Graham Hudson (British Telecom U.K.) former JPEG chair and founder of the effort which became JPEG. The author regrets that space does not permit recognition of the many other individuals who contributed to JPEG's work.

Thanks to Majid Rabbani of Eastman Kodak for providing the example in section 7.3.

The author's role within JPEG has been supported in a great number of ways by Digital Equipment Corporation

References

1. Adobe Systems Inc. *PostScript Language Reference Manual.* Second Ed. Addison Wesley, Menlo Park, Calif. 1990
2. Digital Compression and Coding of Continuous-tone Still Images, Part 1, Requirements and Guidelines. ISO/IEC JTC1 Draft International Standard 10918- 1, Nov. 1991.
3. Digital Compression and Coding of Continuous-tone Still Images, Part 2, Compliance Testing. ISO/IEC JTC1 Committee Draft 10918- 2, Dec. 1991.
4. Encoding parameters of digital television for studios. CCIR Recommendations, Recommendation 601, 1982.
5. Howard, P.G., and Vitter, J.S. New methods for lossless image compression using arithmetic coding. Brown University Dept. of Computer Science Tech. Report No. CS- 91- 47, Aug. 1991.
6. Hudson, G.P. The development of photographic videotex in the UK. In *Proceedings of the IEEE Global Telecommunications Conference, IEEE Communication Society,* 1983, pp. 319- 322.
7. Hudson, G.P., Yasuda, H., and Sebestyén, I. The international standardization of a still picture compression technique. In *Proceedings of the IEEE Global Telecommunications Conference, IEEE Communications Society,* Nov. 1988, pp. 1016- 1021.
8. Huffman, D.A. A method for the construction of minimum redundancy codes. In *Proceedings IRE,* vol. 40, 1962, pp. 1098- 1101.
9. Léger, A. Implementations of fast discrete cosine transform for full color videotex services and terminals. In *Proceedings of the IEEE Global Telecommunications Conference, IEEE Communications Society,* 1984, pp. 333- 337.
10. Léger, A., Omachi, T., and Wallace, G. The JPEG still picture compression algorithm. In *Optical Engineering,* vol. 30, no. 7 (July 1991), pp. 947- 954.
11. Léger, A., Mitchell, M., and Yamazaki, Y. Still picture compression algorithms evaluated for international standardization. In *Proceedings of the IEEE Global Telecommunications Conference, IEEE Communications Society,* Nov. 1988, pp. 1028- 1032.
12. Lohscheller, H. A subjectively adapted image communication system. *IEEE Trans. Commun.* COM- 32 (Dec. 1984), pp. 1316- 1322.

13. Office Document Architecture (ODA) and Interchange Format, Part 7: Raster Graphics Content Architectures. ISO/IEC JTC1 International Standard 8613- 7.

14. Pennebaker, W.B., JPEG Tech. Specification, Revision 8. Informal Working paper JPEG- 8-R8, Aug. 1990.

15. Pennebaker, W.B., Mitchell, J.L., et. al. Arithmetic coding articles. *IBM J. Res. Dev.*, vol. 32, no. 6 (Nov. 1988), pp. 717- 774.

16. Rao, K.R., and Yip, P. *Discrete Cosine Transform- - Algorithms, Advantages, Applications*. Academic Press, Inc. London, 1990.

17. Standardization of Group 3 facsimile apparatus for document transmission. CCITT Recommendations, Fascicle VII.2, Recommendation T.4, 1980.

18. Wallace, G.K. Overview of the JPEG (ISO/CCITT) still image compression standard. Image Processing Algorithms and Techniques. In *Proceedings of the SPIE*, vol. 1244 (Feb. 1990), pp. 220- 233.

19. Wallace, G., Vivian, R,. and Poulsen, H. Subjective testing results for still picture compression algorithms for international standardization. In *Proceedings of the IEEE Global Telecommunications Conference. IEEE Communications Society,* Nov. 1988, pp. 1022- 1027.

Biography

Gregory K. Wallace is currently Manager of Multimedia Engineering, Advanced Development, at Digital Equipment Corporation. Since 1988 he has served as Chair of the JPEG committee (ISO/IEC JTC1/SC2/WG10). For the past five years at DEC, he has worked on efficient software and hardware implementations of image compression and processing algorithms for incorporation in general- purpose computing systems. He received the BSEE and MSEE from Stanford University in 1977 and 1979. His current research interests are the integration of robust real- time multimedia capabilities into networked computing systems.

APPENDIX III

Attached is an excerpt from Pennebaker & Mitchell, *JPEG: Still Image Data Compression Standard*, Van Nostrand Reinhold (1993), at Chapter 7.4, pp. 105-109.

Table 7-3. Vertically-interleaved data units

Component block	*Tq*	*PRED*	MCU
Scan1:			
Y1	00	0	1
Y3	00	DC_{Y_1}	1
Cb1	01	0	1
Cr1	01	0	1
Y2	00	DC_{Y_3}	2
Y4	00	DC_{Y_2}	2
Cb2	01	DC_{Cb_1}	2
Cr2	01	DC_{Cr_1}	2
Y5	00	DC_{Y_4}	3
Y7	00	DC_{Y_5}	3
Cb3	01	DC_{Cb_2}	3
Cr3	01	DC_{Cr_2}	3
Y6	00	DC_{Y_7}	4
Y8	00	DC_{Y_6}	4
Cb4	01	DC_{Cb_3}	4
Cr4	01	DC_{Cr_3}	4

components were present in a scan, it might be tempting to renormalize and interleave as if $H_1 = 2$, $V_1 = 1$ and $H_3 = 1, V_3 = 1$. This is not allowed, however.

7.4 Marker definitions ◑

Each marker segment begins with one or more byte-aligned X'FF' (hexadecimal FF) bytes and a non-zero one-byte "marker code" that identifies the function of the segment. JPEG terms the marker code and its X'FF' prefix a marker. Marker segments and entropy-coded segments always contain an integer number of bytes.

In generating the entropy-coded segments, X'FF' bytes are occasionally created. To prevent accidental generation of markers in the entropy-coded segments, each occurrence of a X'FF' byte is followed by a "stuffed" zero byte. Marker segments have known lengths, and therefore may contain accidental markers. Stuffing of zero bytes is not done in marker segments.

All marker segments and entropy-coded segments are followed by another marker. In Huffman coding one-bits are used to pad the entropy-coded data to achieve byte alignment for the next marker. Because a leading one-bit sequence is always a prefix for a longer code, padding with

Table 7-4. Definition of SOF_n markers

Name	Code	Length	Category
Nondifferential Huffman-coding frames:			
SOF_0	X'FFC0'	V	Baseline DCT
SOF_1	X'FFC1'	V	Extended sequential DCT
SOF_2	X'FFC2'	V	Progressive DCT
SOF_3	X'FFC3'	V	Lossless (sequential)
Differential Huffman-coding frames:			
SOF_5	X'FFC5'	V	Differential sequential DCT
SOF_6	X'FFC6'	V	Differential progressive DCT
SOF_7	X'FFC7'	V	Differential lossless
Nondifferential arithmetic-coding frames:			
SOF_9	X'FFC9'	V	Extended sequential DCT
SOF_{10}	X'FFCA'	V	Progressive DCT
SOF_{11}	X'FFCB'	V	Lossless (sequential)
Differential arithmetic-coding frames:			
SOF_{13}	X'FFCD'	V	Differential sequential DCT
SOF_{14}	X'FFCE'	V	Differential progressive DCT
SOF_{15}	X'FFCF'	V	Differential lossless

V – Variable length with known structure.

one-bits cannot create valid Huffman codes that might be decoded before the marker is identified. In arithmetic coding the entropy-coded segment is terminated at a byte boundary in a way that guarantees that the next marker will be detected before decoding is complete. For either entropy coder, a marker immediately following the entropy-coded segment can provide information needed to terminate decoding.

The markers fall into two categories: those without parameters and those followed by a variable-length sequence of parameters of known structure. For those markers with parameters, the first parameter is a two-byte parameter giving the length (in bytes) of the sequence of parameters. This length includes the length parameter itself, but excludes the marker that defines the segment.

The rest of this section contains a brief summary of markers and their function. Table 7-4 lists the start of frame (SOF_n) markers. Differential frames referred to there are used only within a hierarchical progression. All other markers are listed in Table 7-5. Additional information about markers can be found in Annex B of the DIS.

Na
AP
CO
DA
DH
DH
DN
DQ
DR
EO
EX
JP(
JP(
RE
RS
SO
SO
TE
N -
U -
V -

APP_n: 1
tion use
applicat
data ar
reader
APP_n m
conflicti
be need
applicat

COM:
It may
specifie

DAC: T
defines
ability

Table 7-5. Definition of non-SOF$_n$ markers

Name	Code	Length	Category
APP$_n$	X'FFE0'		
	- X'FFEF'	V	Reserved for application use
COM	X'FFFE'	V	Comment
DAC	X'FFCC'	V	Define arithmetic conditioning table(s)
DHP	X'FFDE'	V	Define hierarchical progression
DHT	X'FFC4'	V	Define Huffman table(s)
DNL	X'FFDC'	4	Define number of lines
DQT	X'FFDB'	V	Define quantization table(s)
DRI	X'FFDD'	4	Define restart interval
EOI	X'FFD9'	N	End of image
EXP	X'FFDF'	3	Expand reference image(s)
JPG	X'FFC8'	U	Reserved for JPEG extensions
JPG$_n$	X'FFF0'		
	- X'FFFD'	U	Reserved for JPEG extensions
RES	X'FF02'		
	- X'FFBF'	U	Reserved
RST$_m$	X'FFD0'		
	- X'FFD7'	N	Restart with modulo 8 counter *m*
SOI	X'FFD8'	N	Start of image
SOS	X'FFDA'	V	Start of scan
TEM	X'FF01'	N	For temporary use in arithmetic coding

N - No length or parameter sequence follows.
U - Undefined at this time.
V - Variable length with known structure.

APP$_n$: The APP$_n$ (Application) marker segments are reserved for application use. Since these segments may be defined differently for different applications, the JPEG DIS states that they should be removed when the data are transferred between application environments. However, the reader should be aware that JPEG did not make the removal of these APP$_n$ marker segments mandatory. Therefore, if two applications require conflicting usage of an APP$_n$ marker segment, checking of the data may be needed when JPEG compressed data is imported or exported from one application to the other.

COM: The COM (Comment) marker segment is intended for text fields. It may not contain information that could affect decoding. JPEG has not specified the character set to be used in this segment.

DAC: The DAC (Define Arithmetic coding Conditioning) marker segment defines one or more tables of parameters for the conditioning of the probability estimates used in the arithmetic coding procedures.

DHP: The DHP (Define Hierarchical Progression) marker segment is used to signal the parameters identifying the dimensions, components, and relative sampling of the final image of the hierarchical progression. It uses the same syntax as the frame header (see section 7.5) and must precede the first frame in the progression.

DHT: The DHT (Define Huffman Table) marker segment defines one or more Huffman tables.

DNL: The DNL (Define Number of Lines) marker segment provides a mechanism for defining or redefining the number of lines in the image at the end of the first scan. The two-byte length (always 4) is followed by a two-byte value giving the number of lines in the frame. The number of lines specified in the DNL marker segment must be consistent with the number of lines and the sampling factors already specified in the frame header and with the number of lines already decoded. Thus, unless the value specified in the frame header is zero, the number of lines specified in the DNL marker segment must be less than or equal to the frame header value, and in all cases must be a value somewhere within the range of the last row of MCU decoded. Note that if the data are directly written to an image buffer, spurious data may be written to the buffer if rows of samples are padded in this last MCU row.

DQT: The DQT (Define Quantization Table) marker segment defines one or more quantization tables. Quantization tables used in a progression should not be changed between scans of that progression. The DQT marker has meaning only for DCT-based algorithms.

DRI: The DRI (Define Restart Interval) marker segment provides the mechanism for setting and resetting the restart interval. The two-byte length (always 4) is followed by a two-byte value giving the number of MCU in the restart interval. The restart interval is set to zero at the start of an image. When the restart interval is zero, the restart function is disabled.

EOI: The EOI (End Of Image) marker terminates the JPEG compressed data stream.

EXP: The EXP (Expand) marker segment signals the expansion of the spatial resolution of the reference data needed by the hierarchical differential frame that follows. It is followed by a two-byte length (always 3) and an eight-bit integer that has one of the values X'10', X'01' or X'11'. X'10' signals a horizontal expansion by a factor of 2, X'01' signals a vertical expansion by a factor of 2, and X'11' signals both horizontal and vertical expansion by a factor of 2.

JPG; JPG$_n$: The JPG and JPG$_n$ marker segments are reserved for future JPEG extensions.

RES: The RES markers are reserved for future JPEG extensions.

RST$_m$: The RST$_m$ (Restart) marker is appended to the compressed data between restart intervals. This provides a unique byte-aligned code that can be located by scanning the compressed data. A three-bit field (m) in this marker code provides a modulo-8 restart interval count. The modulo count of the first RST$_m$ code in a scan is zero.

SOF$_n$: An SOF$_n$ (Start Of Frame) marker begins a frame header. The variable-length SOF$_n$ marker segment gives the frame parameters that apply to all scans within the frame. The particular value of n in SOF$_n$ identifies the mode of compression and the entropy coder used within the frame.

SOI: The SOI (Start Of Image) marker begins the compressed data stream. Note that the SOI marker can be used to detect problems with bit order, bit sense, and bit alignment of the data.

SOS: The SOS (Start Of Scan) marker begins a scan header. The scan header is always followed immediately by entropy-coded data for the scan. The two-byte length gives the number of bytes of scan parameters.

TEM: The TEM marker is reserved for temporary private use by arithmetic encoders. It is used to signal temporarily an unresolved carry-over, so that postprocessing can carry out the resolution off-line. It may not occur in JPEG compressed data streams.

The DHT, DAC, DRI, DQT, COM, and APP$_n$ marker segments may be inserted into the compressed data before the frame and scan headers. They may be used in any order and as many times as desired. If the DHT and DQT marker segments follow the start-of-image marker (SOI), they may be followed immediately by the end-of-image (EOI) marker. This is the JPEG abbreviated format for table specification data. COM and APP$_n$ marker segments may also be used in this abbreviated sequence. The DAC and DRI may appear but have no function, because the next SOI will set default values.

7.4.1 Structure of the compressed data stream ◑

From the definitions of the segments in the previous section the marker and segment structure of a nonhierarchical encoder output might be as shown in Figure 7-5. Note:

1. The marker terminating the final restart interval of the first scan may be preceded by the DNL marker.
2. The last entropy-coded segment in a scan is not followed by an RST$_m$ marker.
3. If arithmetic coding is used, the DAC marker segment is used instead of the DHT marker segment.